Aggregation and Grouping – Key Points

- aggregation operators apply an operation to a group of rows
  - e.g. COUNT, SUM, MAX
- grouping allows rows to be grouped together so that an aggregation operator can be applied to each group instead of (only) the table as a whole
  - GROUP BY to specify column(s) used for grouping
  - HAVING allows the selection of only certain groups
  - ORDER BY sorts the rows in the result
- be aware of how NULL is handled

Aggregation

Aggregation operators apply an operation to a group of rows.

operator	notes
SUM(s)	operator is applied to the values in the column *s*
AVG(s)	
MIN(s)	
MAX(s)	
COUNT(s)	
COUNT(*)	computes the number of rows

Duplicates are included, unless DISTINCT is used.
- e.g. COUNT(DISTINCT s)

Aggregation

```
SELECT COUNT(*) AS numMovies
FROM MOVIES
WHERE year >= 2000
```
the number of movies released in 2000 and after

```
SELECT MIN(length) AS shortest,
       MAX(length) AS longest
FROM MOVIES
WHERE year = 2012
```
the shortest and longest movies released in 2012

```
SELECT COUNT(DISTINCT year)
FROM ( MOVIES M JOIN STARSIN SI
          ON M.title=SI.movieTitle ) JOIN
       MOVIESTAR MS ON SI.starName=MS.name
WHERE MS.name = 'Harrison Ford'
```
the number of different years in which Harrison Ford has starred in a movie

What to Count

```
SELECT COUNT(*)
FROM SAILOR JOIN RESERVATION USING (Sid)
```
- counts the number of reservations

```
SELECT COUNT(Sid)
FROM SAILOR JOIN RESERVATION USING (Sid)
```
- counts the number of reservations (Sid is PK, so cannot be NULL)

```
SELECT COUNT(Sname)
FROM SAILOR JOIN RESERVATION USING (Sid)
```
- counts the number of reservations made by sailors whose Sname is not NULL

```
SELECT COUNT(DISTINCT Sid)
FROM SAILOR JOIN RESERVATION USING (Sid)
```
- counts the number of different sailors who have made reservations

```
SELECT COUNT(DISTINCT Sname)
FROM SAILOR JOIN RESERVATION USING (Sid)
```
- counts the number of different names of sailors who have made reservations

Grouping

Grouping allows rows to be grouped together so that an aggregation operator can be applied to each group instead of (only) the table as a whole.

```
SELECT studioName, AVG(length)
FROM MOVIES
GROUP BY studioName
```

for each studio, the average length of that studio's movies

- GROUP BY groups rows which have the same value for the specified attribute(s)
- aggregation operations in the SELECT clause are applied to each group
- other columns in the SELECT clause can only be those used for grouping
  - or those dependent on the PK when grouping columns include the PK

Grouping

- use the primary key when you want to uniquely identify things

```
SELECT deptname, COUNT(*)
FROM EMPLOYEE NATURAL JOIN DEPARTMENT
GROUP BY deptnum

SELECT deptname, COUNT(*)
FROM EMPLOYEE NATURAL JOIN DEPARTMENT
GROUP BY deptname
```

- in the second query, different departments which happen to have the same name will be combined into a single count – this is probably not what is desired

Grouping

```
SELECT studioName, AVG(length)
FROM MOVIES
WHERE year = 2000
GROUP BY studioName
```

for each studio, the average length of the movies that studio released in 2000

- WHERE selects individual rows before grouping is applied

```
SELECT studioName, AVG(length)
FROM MOVIES
GROUP BY studioName
HAVING COUNT(*) > 2
```

for those studios which have more than two movies, the average length of that studio's movies

- HAVING selects groups
- aggregation operators in HAVING apply to just the one group
- HAVING condition must make sense with one row per group
  - columns in HAVING must be aggregated or grouping columns or GROUP BY includes PK

Examples

```
SELECT deptnum,COUNT(ssn)
FROM EMPLOYEE
GROUP BY deptnum
```

the number of non-NULL ssns in each department...which, since ssn is EMPLOYEE's PK, is equivalent to the number of employees in each department

```
SELECT deptnum
FROM EMPLOYEE
GROUP BY deptnum
HAVING COUNT(ssn) > 2
```

the departments with more than two non-NULL ssns...i.e. more than two employees

Grouping

What order are the clauses in the following query applied? What does it do?

```
SELECT Rating, COUNT(*)
FROM SAILOR
WHERE Age > 30
GROUP BY Rating
HAVING COUNT(*) > 1
ORDER BY Rating
```

FROM SAILOR
→ start with all of the sailors

WHERE Age > 30
→ just the sailors who are over 30

GROUP BY Rating
→ the sailors over 30, grouped by rating

HAVING COUNT(*) > 1
→ just those groups with more than one row (i.e. at least two) = just those ratings with at least two sailors over 30

ORDER BY Rating
→ list the results in increasing order by rating

SELECT Rating, COUNT(*)
→ for the ratings with at least two sailors over 30, the number of sailors over 30 with that rating

Grouping

```
SELECT a1,a2,...,an
FROM R
WHERE C
GROUP BY b1,b2,...,bm
HAVING H
ORDER BY c1,c2,…,ck
```

- build the table in FROM
- pick rows matching the WHERE condition
- group those rows into groups with the same values for all of b1,b2,…,bm
- pick the groups matching the HAVING condition
  - *H* can only involve things for which there is only a single value per group – when there's aggregation or GROUP BY columns, or other columns from the same table if GROUP BY involves a primary key
  - one result row per group
- order the group rows by b1,b2,…,bm
- include just the columns a1,a2,…,an for each group row

NULL

NULL is ignored in aggregation – it does not count towards SUM, etc.
- but COUNT(*) counts all rows, even if they contain NULLs

NULL is treated as an ordinary value when forming groups.
- groups may have NULL values in grouping attributes, and those will be separate groups from those without NULLs

The value of an aggregation computed on an empty bag is NULL, except for COUNT (which returns 0).

Examples

```
SELECT deptnum, COUNT(*) AS numemployees
FROM EMPLOYEE
GROUP BY deptnum
```
number of employees in each department

```
SELECT deptnum, COUNT(*) AS numemployees
FROM EMPLOYEE
GROUP BY deptnum
HAVING COUNT(*) > 5
```
number of employees in each department, for those departments with more than 5 employees

```
SELECT projnum, COUNT(*) AS totalassignments,
       COUNT(hours) AS assignmentswithhours
FROM WORKS_ON
GROUP BY projnum
```
for each project, the total number of work assignments and the number of work assignments with hours assigned for that project

```
SELECT deptnum, SUM(salary) AS totalsalary, AVG(salary) AS avgsalary,
       MIN(salary) AS minsalary, MAX(salary) AS maxsalary
FROM EMPLOYEE
GROUP BY deptnum
```
for each department, the sum, average, min, and max salaries for employees in that department

Examples

```
SELECT empssn, COUNT(*) AS numassignments
FROM WORKS_ON
WHERE hours > 20
GROUP BY empssn
```

for each employee who works at least 20 hours on a project, the number of work assignments over 20 hours each – note that the WHERE is before the grouping, so any employees that don't work more than 20 hours on a project are not represented when groups are formed

```
SELECT empssn, COUNT(*) AS numassignments
FROM WORKS_ON
GROUP BY empssn
HAVING SUM(hours) > 20
```

for each employee working at least 20 hours combined on projects, the number of work assignments for that employee

```
SELECT deptnum, COUNT(*) AS numemployees
FROM EMPLOYEE
WHERE deptnum > 5
GROUP BY deptnum
HAVING COUNT(*) > 3
```

both of these queries produce the same result: the number of employees in each department, for departments numbered higher than 5 and with more than three employees each

```
SELECT deptnum, COUNT(*) AS numemployees
FROM EMPLOYEE
GROUP BY deptnum
HAVING COUNT(*) > 3 AND deptnum > 5
```

they differ in strategy but in this case hae the same result

Examples

```
SELECT EMPLOYEE.ssn, EMPLOYEE.fname, EMPLOYEE.lname, EMPLOYEE.salary,
       COUNT(WORKS_ON.projnum) AS numprojects
FROM EMPLOYEE
 JOIN WORKS_ON ON EMPLOYEE.ssn = WORKS_ON.empssn
GROUP BY EMPLOYEE.ssn
HAVING COUNT(WORKS_ON.projnum) > 1
```

work assignments are grouped by employee, then the groups with more than one non-NULL projnum are picked – for each employee working on at least two projects, the number of projects they work on

```
SELECT sex, relationship, COUNT(*) AS numdependents
FROM DEPENDENT
GROUP BY sex, relationship
```

the number of dependents by sex and relationship to the associated employee

```
SELECT projnum, COUNT(DISTINCT empssn) AS numdistinctemployees
FROM WORKS_ON
GROUP BY projnum
HAVING numdistinctemployees >= 3
```

for each project with at least three different employees working on it, that count of employees

Examples

- Show how many employees work in each department. Include both the department number and name.
- Find the average salary of female employees in each department. List departments by number.
- Find departments where the average employee salary exceeds $50,000. List departments by name.
- Show how many employees fall into each combination of department and sex. List departments by name.
- For each project, list the project's name, location, and the total number of hours employees work on that project.
- For each department, show how many work assignments its employees are involved in overall and how many distinct employees that represents. Include both the department number and name.
- For every employee, show how many dependents they have on file, including employees with none.
- List the names of departments that have more than three employees working on projects located in Houston.